

AP-
: implementation file

```
#ifndef  
#define  
#undef  
static char  
#endif
```

```
////////////////////////////////////  
// APostOffice
```

```
IMPLEMENT_DYNCREATE(APostOffice, APostOffice)  
APostOffice::APostOffice()  
{  
    EnableAutomation();
```

keep the application.
object is active, th

Automation
App.

```
(NULL, "PostOffice",
```

```
A.  
{  
    // When the last object is released, the destructor calls  
    // the destructor.  
    // "PostOffice", MB_OK);  
    AfxOleU.
```

```
void APostOffice::OnFinalRelease()  
{
```

```
    // When the last reference to the object is released,  
    // OnFinalRelease is called. This method automatically  
    // deletes the object. This method must be called before  
    // the object is released.
```

```
    MessageBox(NULL, "OnFinalRelease", "PostOffice", MB_OK);
```

```
    CPostOffice::OnFinalRelease();
```

```
    CPostOffice, CCmdTarget)  
    CPostOffice)  
    the Classwizard will add a
```

EN.

BEGIN_

DI.

```
    CPostOffice, CCmdTarget)  
    CPostOffice)  
    "PostOffice", GetName, VT_BSTR,  
    Page 1
```

BEST AVAILABLE COPY

EXHIBIT A

```
TON(APostOffice, "GetLo
```

VTS_VARIANT

```
APostOffice, "Start", S
PostOffice, "Stop", Stop
PostOffice, "Create", Crea
PostOffice, "GetComputer", G
PostOffice, "Handshake", Hands
```

VTS_

```
"MapCERR", MapCERR, VT_
"MapCOUT", MapCOUT, VT_BU
"Close", Close, VT_BOOL, Vi
```

D1

///

END_DISPATCH_M

```
// Note: we add sup
// from VBA. This
// disinterface in t
```

support typesafe binding
is attached to the

C57D0B-CE64-11CF-8BA

const IID IID_IAPostO

0b, 0xce64, 0x11cf,

0x1a, 0xe7, 0x8e } };

E_MAP(APostOffice, C

E_PART(APostOffice, _

-8BAF-0000E81AE78E}

PostOffice, "Eliza.Postu

0x1a, 0xe7, 0x8e)

0x11cf,

////////////////////////////////////

BOOL APC

ie, long Timeout)

{

retu

Timeout);

}

BOOL APostOffice::

{

return CPostO

}

STR APostOffice::GetName()

CString strResult=CPost
return strResult.AllocSys

PostOffice::GetLocalIds(const VA

cept only array declared as

(VT_ARRAY|VT_BYREF|VT_I2),

* psa=*Ids.parray;

ayAccessData(psa, (void**).

GetLocalIds(pData, Max);

(psa);

}

BOOL APost

APOS

ffice::StartTraffic();

Timeout)

hTraffic(Timeout);

BSTR APosto

{

// TODO
char buf1;
DWORD leng=
GetComputerNa
CString strResu
return strResult..

code here
1="";

ffice::Handshake(LPC

const VARIANT FAR& Ids,

cept only array decla.

(VT_ARRAY|VT_BYREF|V.
* psa=*Ids.pparray;

ayAccessData(psa,(vo
ffice::Register(Name,
(psa);

SE;

}

BOOL APosto

{

return

short meth)

eth);

}

BOOL APostOffice::Map

{

return CPostOffi

header file APC.

```
////////////////////////////////////  
// A  
class _ : public CCmdTarget,  
{  
    DECL
```

```
// Attributes  
public:
```

```
    operations  
    ic:
```

```
    APostOffice();  
    virtual ~APostOffice(),
```

used by dynamic creation

```
    wizard generated virt  
    TUAL(APostOffice)  
    FinalRelease();
```

pr

functions

ard will add and remov

```
//}},
```

```
DECLARE_  
DECLARE_OL
```

```
// Generated O
```

```
//{{AFX_DISPATCH
```

```
afx_msg BSTR GetNa
```

```
afx_msg short GetLoc
```

```
afx_msg BOOL Start();
```

```
afx_msg BOOL Stop(long
```

```
afx_msg BOOL Create(LPCTS,
```

```
afx_msg BSTR GetComputer();
```

```
afx_msg BOOL Handshake(LPCTST,
```

```
);
```

```
    BOOL MapCErr(short rcvr,
```

```
    BOOL MapCOut(short rcvr, sh
```

```
    Close(long Timeout);
```

```
    TCH
```

```
    MAP()
```

```
    MAP()
```

```
///
```

```
////////////////////////////////////
```

CPo.

```
class CM-
```

```
{  
    CPostOf  
    WORD rcvr  
    CString lin
```

```
public:
```

```
    CMsgStream(CPostOf  
                : po(_po), rc
```

```
    meth)
```

```
    buf:
```

```
        underflow() { return  
        overflow(int c)
```

```
        r(c);
```

```
        and(CMsg(0,rcvr,meth,L
```

```
        ";
```

```
};
```

```
//=====
```

```
CRCvr::CRCvr(WORD _id  
              : id(_id), po(_
```

```
{
```

```
    cerr::cerr;  
    cout::cout;
```

```
    CRCvr()
```

```
    fice(int nLcl, int iLcl,  
        r>(nLcl,iLcl), clients(nRe  
        L), cout(NULL)
```

```
    nt(NULL,TRUE,TRUE,NULL);
```

```
CPostOf.  
{
```

```
    bEn
```

```
ngleobject(hThreadEnded,  
hThreadEnded);
```

```
rcvr, WORD meth)
```

```
this,rcvr,meth);  
{
```

```
}  
return TRUE;
```

```
CPostOffice::MapCout\
```

```
delete cout;  
cout=cout=new CMsgStre  
for(int i=0; i<GetSize();  
CRCvr* pr=GetAt(i);  
if(pr->IsLocal())  
pr->cout=cout;
```

```
TR lpszName, DWORD tim
```

```
ne);
```

```
int CPostOffi\
```

```
{  
for(int i=0; i<GetSize();  
if(!  
return -1;  
}
```

```
CPostOffice::GetLocalIds\
```

```
int cnt=0;  
for(int i=0; i<GetSize();  
if(GetAt(i)->IsLocal\  
*Ids++ = GetAt\  
cnt;
```

```
ndTo(int handle, CMsg& msg,
```

```
) && (handle<GetSize()) &
```

```
);
```

```
class __de
```

```
public CRCvr  
page 2
```

Cr

-lient;

parent, WORD id, CPosto

cli

{
}

q);

BOOL IsLoc
void Idle()

Office::Register(LPC

WORD Ids, int Size)

create new client
client* pcli=new CPipe
establish a link to
Open(Computer,Name))
0;
i);
ze; i++)
mRcvr(pcli,Ids[i],*th
e());

}

//
//
//

k

BOOL CPostOffic
{

bEndThread=
ResetEvent(hi
for(int i=0; i<
GetAt(i)-
DWORD id;
HANDLE hThread=Creat
return hThread!=NULL;
return AfxBeginThread(Th
return _beginthread((void)

c,LPVOID(this),0,&id);

:
VOID(this))!=-1;

ce::StopTraffic(DWORD dwT

=TRUE;
tForSingleObject(hThreadE

0;

i=0; i<GetSize(); i++)
tAt(i)->OnStop();

}

ULONG
{

ID param)

Cr

*)param;
Page 3

```
CP  
->bEndThread) {  
    wait for next message o  
    *pMsg=ppo->server.Read(p  
    =NULL) {  
        ppo->Send(*pMsg);  
        delete pMsg;
```

```
        i=0; i<ppo->GetSize();
```

```
    }  
    return
```

```
    BOOL CPostOffice::
```

```
    {  
        return server
```


CPU

H

#inc

#include

class CMsgSt
class CPostOff

Abstract clas

declspec(dllexport,

id;
ffice& po;

PostOffice;

cerr, cout;

pu

& _po);

WORD
CPostO.

po; }

virtual BO

virtual void On
virtual void Idl
virtual void OnSto,

virtual BOOL Receive(C

0;

lspec(dllexport) CPostOffic

server;
neClient> clients;
t;

*cout;

int iLcl=10, int nRem=20,

BOG.

WORD timeout=1000) ;
e 1

CP

```
{ return server.GetNe
```

```
WORD Ids, int max);  
R Name, LPCSTR Compute.
```

```
return SendTo(GetHandl  
Msg& msg);
```

```
Bool  
Bool  
eth);  
h);
```

```
BOOL Sta  
BOOL Stop, TTE);
```

```
BOOL Close(DWC
```

```
re:
```

```
BOOL bEndThread;  
OLE hThreadEnded;  
ULONG __stdcall Th
```

ORD m, LPCSTR str)

```
    if (s  
    else da  
    .str, size);
```

n::CMsg(CMsg& m)

```
    sndr=m.sndr;  
    rcvr=m.rcvr;  
    meth=m.meth;  
    size=m.size;
```

```
    memcpy(data=new BYTE[s  
    NULL;
```

size;

};

BOOL CM
{

```
    Msg  
    DWORD  
    delete
```

```
    if(ReadF  
    (GetLastError()==EK  
    cbRead,NULL) ||
```

```
    sndr=  
    rcvr=hu  
    meth=hdr.  
    size=hdr.s.  
    if(!size)
```

```
        return  
    data=new BYTE[s  
    if(ReadFile(hFile,  
    ror()==ERROR_MORE_DATA))  
        return TRUE;
```

FALSE;

hFile, HANDLE hStop)

ULL;

TRUE, FALSE, NULL);

e.

```
ovl;  
events[0];  
hEvent);  
hdr, sizeof(hdr), &cbRead,  
objects(2, events, FALSE,  
r;
```

```
1;t);  
ze, &cbRead, &ovl);  
)==WAIT_OBJECT_0;
```

```
ok=waitForMultipleObj  
}  
CloseHandle(events[0]);  
return bok;
```

```
e(HANDLE hFile)
```

```
sizeof(hdr), &cbWritten,
```

```
retur. &cbWritten, NULL);  
}
```

```
class
```

```
{  
    WORD  
    WORD  
    WORD  
    WORD  
    BYTE da
```

```
bytes)
```

```
public:
```

```
    CMsg(WORD s, WORD  
    L) {}  
    CMsg(WORD s, WORD r, W  
    (CMsg&);
```

```
, meth(m), size(0),
```

```
    ~CMsg() { delete []
```

```
        const { return sndr;  
        const { return rcvr;  
        const { return meth;  
        { return size;  
        return data; }
```

```
        return PCHAR(data); }  
        return PWORD(data); }  
        return PDWORD(data); }
```

```
//  
CMsg(  
    BOOL Re  
    BOOL Rea  
    BOOL Write
```

```
    sndr=rcvr=meth=0; }
```

```
};
```

```
};
```

```
endif
```

```
CPipe
```

```
CPipeClient::~CPipeClient()
```

```
if(hPipe!=INVALID_HANDLE_VALUE)
    CloseHandle(hPipe);
```

```
CPipeClient::Open(LPCTSTR
```

```
try to perform local
ComputerName[256];
Length=sizeof(ComputerName);
strcpy(ComputerName,
```

```
Computer[0]&&1strcmp
```

```
 "\\pipe\\" + Name;
```

```
/*
```

```
[SECURITY_ATTRIBUTES
```

```
InitSecurityAttributes(
SetSecurityDescriptorDa
SECURITY_DESCRIPTOR
sa.nLength=sizeof(SECURITY_DESCRIPTOR);
sa.lpSecurityDescriptor=sd;
sa.bInheritHandle=FALSE;
```

```
*/
```

```
// open a connection
while(1) {
```

```
hPipe=CreateFile(
    Path,
    GENERIC_READ|GENERIC_WRITE,
    0,
    NULL,
    OPEN_EXISTING,
    FILE_ATTRIBUTE_NORMAL,
    NULL);
```

```
if(hPipe==INVALID_HANDLE_VALUE)
    break;
if(GetLastError()!=ERROR_PIPE_BUSY)
    break;
if(!NamedPipe(PipeName,20000))
    break;
```

```
return (HANDLE_VALUE);
```

```
}
```

```
BOOL CPipeClient::Open(LPCTSTR
```

```
->write(hPipe);
```

```
ue()
```

```
FALSE, NULL);
```

```
CPipeServer
```

```
closeHa
```

```
BOOL CPipeServer::CMsg
```

```
if(!IsEmpty()) ret
```

```
if(Timeout==0) retur
```

```
if(WaitForSingleObject
```

```
ResetEvent(hPut,
```

```
return TRUE;
```

```
SE;
```

```
ueue::Get()
```

```
TRUE);
```

```
:Get();
```

```
}
```

```
void CPipe
```

```
p)
```

```
CSing.
```

```
TIQueue
```

```
SetEvent(i.
```

```
BOOL CPipeServer::CMsgQu
```

```
CSingleLock sl(&lock
```

```
BOOL bEmpty=TIQueue<C
```

```
return bEmpty;
```

```
pipeServer()
```

```
(10,10)
```

```
reateEvent(NULL, TRUE, FALSE,
```

```
}
```

```
BOOL Cpip
```

```
ame)
```

```

    Name;
    \\pipe\\"+Name;
    thread
    +Stop);
    on();

```

```

CMSs
{
    timeout)
    t)?MsgQueue.Get():NULL;
}

```

```

BOOL CPipeServ
{
    if(!threa...
        retu...
    // global stop
    SetEvent(hEvents...
    // wait for all th...
    DWORD
    MultipleObjects(thre...
    t==WAIT_FAILED || wt...
    for(int i=0; i<thre...
        if(WaitForSin...
    }
    rn FALSE;
    threads.GetSize(); i++,
    e(threads[i]);
}

```

```

BOOL CPi...
{
    DWORD
    HANDLE
    h=CreateThread(N...
    if(h) {
        thre...
        Resume
        return i...
    }
    return FALSE;
    return AfxBeginThread\
    return _beginthread((vo...
    )!=NULL;
    d,4096,LPVOID(this))!=-1,

```

```

eServer::Connection()

```

```

ty...
    DESCRIPTOR psd=(PSECURITY_D...
    MIN_LENGTH];
    ityDescriptor(psd, SECURITY_...
    torDacl(psd, TRUE, NULL, FAL...
    sa;
    =psd;

```

```

*/
// C...
HAN...

```



```

th,
ACCESS_INBOUND | FILE_
TYPE_MESSAGE |
MODE_MESSAGE |
D_INSTANCES,

```

```

//
// BOOL
bConn=ConnectN(
OVERLAPPED,
ovl.hEvent,
ConnectNamedPipe,
DWORD dwErr=Ge
BOOL bConn=FALSE,
if(dwErr==ERROR_PIPE_
bConn=TRUE;
else if(dwErr==ERROR_
// wait for conn
HANDLE evnts[2];
evnts[0]=ovl.hEvent;
evnts[1]=hEventStop;
n=WaitForMultipleObjec
ns connection
WAIT_OBJECT_0;

next thread
n();
this connection...
CMsg; p->Read(hPipe,h

p=new
}
//
Close
Flush
Disconn
CloseHand
return 0;
}

```

H

```
#in
```

```
#include
```

```
#include
```

```
class __declspec
```

```
{
```

```
    HANDLE hPipe;  
    CString Name;  
    CString Path;
```

```
    ~CClient();
```

```
    ~CPipeClient();
```

```
    (LPCSTR lpszComputer,  
     CMsg*);
```

```
cla
```

```
{
```

```
neServer
```

```
    CSt.
```

```
class __
```

```
{
```

```
ue : public TQueue<C
```

```
    CMU  
    HANDL.
```

```
public:
```

```
    CMsgQueue  
    ~CMsgQueue;  
    CMsg* Get();  
    void Put(CMsg*);  
    BOOL IsEmpty();  
    BOOL Wait(DWORD T1,
```

```
    CMsgQueue;
```

```
    hEventStop;  
    HANDLE> threads;
```

```
er();
```

```
zName);
```

```
ut=INFINITE);
```

```
=INFINITE);
```

```
; }
```

```
stdcall ConnectionThre
```

```
PipeServer*)param)->Con.
```

```
};
```

```
#endif
```

Pos.
Defines the initializa.

```
#if  
#def  
#undef  
static ch  
#endif
```

```
/////////////////////////////////  
// CPostOfficeApp
```

```
BEGIN_MESSAGE_MAP(CPostOfficeApp  
    //{{AFX_MSG_MAP  
    // NOTE -  
    // DO NOT  
    //}}AFX_MSG_MAP  
    MAP()
```

' remove mapping macros here,
blocks of generated code!

```
/////////////////////////////////  
construction
```

```
CPostOfficeApp()
```

construction code here,
significant initialization

```
/////////////////////////////////  
// The
```

```
object
```

```
CPostOfficeApp
```

```
/////////////////////////////////  
// CPostOfficeApp
```

```
BOOL CPostOfficeApp::InitInstance()  
{  
    // Register all COM  
    // OLE libraries  
    COleObjectFactory::RegisterAll()  
  
    return TRUE;  
}
```

...ing. This enables the
applications.

```
/////////////////////////////////  
try points required for in
```

```
CPostOfficeApp::GetObject(REFCLSID rclsid, REFIID riid, void** ppvObject)
```

```
if (SUCCEEDED(AfxGetStaticModuleState(&pModule)))  
    return ClassObject(rclsid, riid, ppvObject, pModule);
```

```
return AfxGetStaticModuleState();
```

Post
registerServer, you can
er(void)

fxGetStaticModuleState
ndateRegistryAll();

}

PC
· Declares the module pa
"CE"
Windows Dynamic Link L
here

Postu
in header file for the ,

'x.h' before including ,

#e.

#includ

main symbols

/////////
// CPostOffi
// See PostOff,
//

////////////////////////////////////

on of this class

class CPostOfficeApp

ic:

CPostOfficeApp();

as

Classwizard generated

_VIRTUAL(CPostOffice

OL InitInstance();

IAL

PostOfficeApp)

the Classwizard will
EDIT what you see in

ons here.
ode !

};

/////////

////////////////////////////////////

Po-
 type library source file
 processed by the Make Typ
 Library (PostOffice.tlb).
 {F-0000E81AE78E}, version

{
 {

//
 [uuid(59C57D0C-CE64-4
 dispinterface
 {

property
 //
 //{{AFX_...
 //}}AFX_...

methods:
 // NOTE - Class.
 // Use extreme
 //{{AFX_ODL_METHODS
 [id(1)] BSTR GetName();
 [id(2)] short GetLocal;
 [id(3)] boolean Start();
 boolean Stop(long T.
 boolean Create(BSTR Na
 BSTR GetComputer();
 boolean Handshake(BSTR Nam

Ids, short

MapCerr(short rcvr, si
 bCOut(short rcvr, sho
 se(long Timeout);

};
 // class informat
 [uuid(59C57D0C-CE64-4
 coclass APostOffice
 {
 [default] dispinter.

APPEND_ODL}}

PC
: type library source

processed by the Make Ty.
Library (PostOffice.tlb).

AF-0000E81AE78E), versio

```
//  
[ uuid(59C57D0C-CE64-40A8-B1AE78E) ]  
dispinter.  
{  
    proper
```

maintain property information
writing this section.

```
//  
//{{AF-  
//}}AFX_
```

methods:

```
// NOTE - Class  
// Use extreme  
//{{AFX_ODL_METHODS  
[id(1)] BSTR GetName()  
[id(2)] short GetLocalI  
[id(3)] boolean Start()  
[id(4)] boolean Stop(long l  
[id(5)] boolean Create(BSTR Na  
[id(6)] BSTR GetComputer();  
[id(7)] boolean Handshake(BSTR Na
```

information here.
section.

Ids, short

```
MapCerr(short rcvr, si  
MapCout(short rcvr, sho  
ase(long Timeout);
```

```
};
```

```
// Class informa
```

```
[ uuid(59C57D0C-CE64-40A8-B1AE78E) ]  
coclass APostOffice  
{
```

```
    [default] dispinter.
```

```
APPEND_ODL}}
```

PC
C++ generated resource

SYMBOLS

2 resource.

#include

#undef APSTUDIO

#ifdef APSTUDIO_IN

TEXTINCLUDE

UDE DISCARDABLE

h\0"

DABLE

""\r\n"

3 TL
BEGIN

"#de
"#def,
"#define
"#define _

"\r\n"

"#if !def,

"#ifdef _WIN

"LANGUAGE 9, _

"#pragma code_pa

"#endif\r\n"

"#include ""res\\Po

sources\r\n"

"#include ""afxres.rc"

"#endif"

CES\r\n"

"

"\n"

"n"

fined(AFX_TARG_ENU)\r\n"

soft visual C++ edited

nts\r\n"

STUDIO_INVOKED

LL) || defined(AFX_TARG_Ei

LA
#prag
#endif

ge 1

PL

WININFO

```
FILE
#ifdef
FILEFL
#else
FILEFLAGS
#endif
FILEOS 0x4L
FILETYPE 0x2L
FILESUBTYPE 0x0L
BEGIN
```

```
BLOCK "StringFi
BEGIN
BLOCK "040904B0"
BEGIN
```

```
VALUE "C
VALUE "Fi
VALUE "Filev
VALUE "Interna
VALUE "LegalCopy
VALUE "LegalTradem
VALUE "OriginalFilen
VALUE "ProductName",
VALUE "ProductVersion",
```

l\0"

f\0"

ibrary\0"

0x409, 1200

```
END
END
```

```
#endif
```

```
#ifndef APSTUDIO_INVOKE
////////////////////////////////
```

Generated from the TEXTIN

```
_AFX_NO_SPLITTER_RESOURCES
_AFX_NO_OLE_RESOURCES
_AFX_NO_TRACKER_RESOURCES
_NO_PROPERTY_RESOURCES
```

```
RESOURCE_DLL) || defined,
```

rc2" // non-Microsoft Vi
standard components

```
#c
```

```
////////////////////////////////
#endif
```

```
re  
-}}++ generated include fi
```

```
w objects
```

```
//  
#ifac  
#ifnde.
```

```
#define _Ar 129  
#define _APS_ 2771  
#define _APS_Ne.  
#define _APS_NEXI.  
#endif  
#endif
```

source file that includes
which will be the pre-comp.
will contain the pre-comp.

Standard
file for standard sys-
include files that are
requently

```
#ifndef _AFX_ // Exclude rarely-used  
#include <afxcore.h> // core and standard compo-  
#include <afxext.h> // extensions  
  
#ifndef _AFX_ // Exclude rarely-used  
#include <afxwin.h> // Windows classes  
#include <afxodl.h> // OLE classes  
#include <afxdisp.h> // COM classes  
#endif // _AFX_NO_OLE
```

```
#ifndef _AFX_NO_DB_SUPPORT  
#include <afxdb.h> // Database classes  
#endif // _AFX_NO_DB_SUPPORT  
  
#ifndef _AFX_NO_DAO_SUPPORT  
#include <afxdao.h> // DAO classes  
#endif // _AFX_NO_DAO_SUPPORT  
  
#ifndef _AFX_NO_MFC_SUPPORT  
#include <afxmfc.h> // MFC classes  
#endif // _AFX_NO_MFC_SUPPORT  
  
#ifndef _AFX_NO_COMMON_SUPPORT  
#include <afxcom.h> // COM classes  
#endif // _AFX_NO_COMMON_SUPPORT
```

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☒ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☒ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☒ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.